uc3m | Universidad Carlos III de Madrid

Grado en Ingeniería en Tecnologías Industriales
2017/2018

Trabajo Fin de Grado

Implementación de un sistema Data Muling Industrial para Arduino

Gabriel Pavón Barrero

Tutor:
Pablo Basanta Val

Leganés, 3-18 octubre, 2018

1. Agradecimientos

A mi padre por su ayuda durante este trabajo y durante toda la carrera.

A mi madre por su cariño, paciencia y sus consejos.

Y a todas las personas que han estado ahí durante estos años.

2. Resumen

En numerosas ocasiones se encuentran entornos de monitorización en los que las condiciones no permiten el uso de redes ad-hoc multisalto. Normalmente se debe a las grandes distancias entre los sensores. Estas redes aisladas se conocen como redes de sensores inalámbricos o WSN (*Wireless Sensor Networks*, tienen una gran autonomía energética y la capacidad de crear redes ad-hoc para comunicarse entre sí.

Una posible solución al problema de la distancia con la red WSN es el uso de un nodo móvil, conocido como "mula de datos", que se desplace físicamente entre las redes, descargando la información cuando se encuentra próximo a los sensores para después enviar dicha información a un nodo pasarela hacia otra red o, directamente, a un servidor que la almacene.

Durante este proyecto se pretende implementar una plataforma tanto a nivel de hardware como de software que actúe como mula de datos. Para ello, se utilizan placas de Arduino, que toman los papeles de los distintos sensores de la red y se estudian las funcionalidades de éstas. Para realizar las comunicaciones entre las placas Arduino se utilizan módulos basados en radiofrecuencia. En esta primera parte, se describe también la arquitectura interna y se analizan otros proyectos en las que se utilizan mulas de datos.

En la segunda parte, se describe el diseño (tanto de hardware como de software) con mayor detalle, estudiando los aspectos más relevantes del prototipo. Se analizan los requisitos que debe cumplir cada tipo de nodo y el protocolo de comunicación entre ellos.

Por último, se incluyen una serie de pruebas para comprobar el rendimiento del sistema, las conclusiones y las líneas de trabajo futuro.

Palabras Clave Arduino, WSN, Data Mule, Módulos RF

Índice

Índice de figuras

3. Introducción

En esta primera sección se describe brevemente el contexto de las telecomunicaciones, así como los objetivos principales de este proyecto.

3.1. Contexto

Los sistemas de comunicación inalámbrica tienen un impacto significativo en la sociedad moderna. Con el descubrimiento de las ondas electromagnéticas surgieron las primeras ideas para comunicarse sin la necesidad de emplear cables. En 1888, Rudolf Hertz, un físico alemán, consigue realizar una transmisión inalámbrica por primera vez [21].

Desde las primeras ondas enviadas y recibidas con éxito, la tecnología que rodea a las telecomunicaciones está en continuo desarrollo. Como muchos otros campos de la ingeniería, las telecomunicaciones vieron un gran avance al asociarse con el ámbito militar. Sin embargo, es cuestión de tiempo que se desarrollen aplicaciones de todo tipo: urbanas, industriales, agrícolas, transporte, etc. La implementación de redes permite recopilar y mover una gran cantidad de datos de manera eficiente, dando lugar al estudio de estos datos y desarrollo de mejoras de forma continuada.

Nuestro mundo está plagado de elementos y sensores electrónicos. Como ejemplo, cualquier oficina actual tiene un termostato, todos los coches tienen docenas de sensores, etc. Los datos que recopilan estos sensores son utilizados por los sistemas locales pero, ¿qué pasaría si pudieramos comunicar y utilizar todos estos datos de forma remota?[23] Aquí entran en juego las redes inalámbricas.

Estas redes inalámbricas pueden, comúnmente, utilizar diferentes formas para comunicarse. Los dos métodos más comunes son las redes ad-hoc, y una infraestructura basada en un nodo central, Figura 1, llamado punto de acceso.

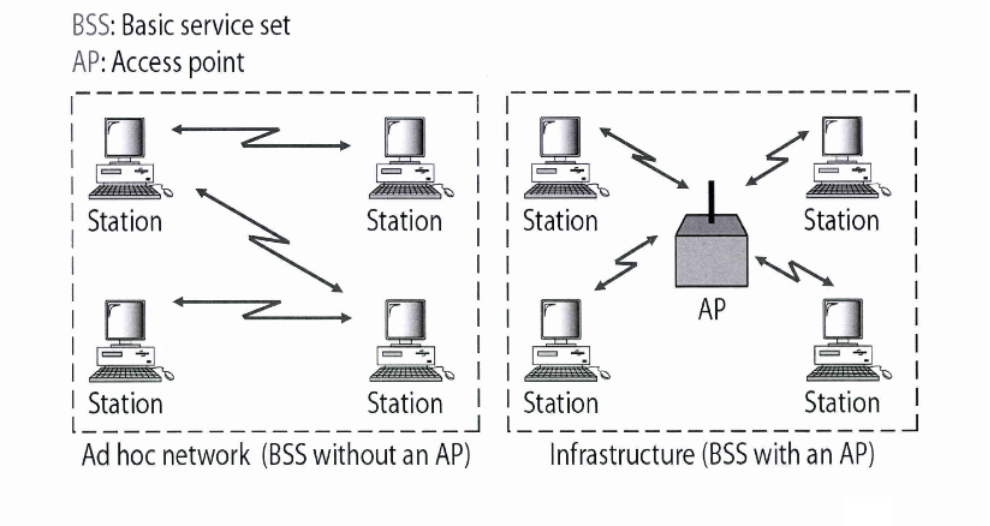

Figura 1: Adhoc Vs Infraestructura[33].

Resultan de especial interés las redes ad-hoc, pues permiten una comunicación continuada con cualquier nodo que forme parte de la red. Se denominan redes *ad-hoc multisalto*. Los nodos se comunican entre si almacenando información, hasta que ésta llega a un nodo pasarela, encargado de llevar esa información a una red o servidor externo. Para lograr tener una red eficiente debe haber una gran cantidad de nodos y que su distribución sea relativamente homogénea.

Sin embargo, esta distribución de abundantes nodos no es siempre posible. Existen entornos donde las distancias son demasiado grandes, encareciendo demasiado el despliegue de redes de este tipo. Se crean así subredes locales específicas para el área que se desea monitorizar[12].

Las subredes locales no son capaces de comunicarse entre si. Entre las posibles soluciones, encontramos el empleo de un nodo móvil o "mula de datos"[27] que se desplace físicamente entre estas subredes. Así, incluyendo un nodo con movilidad, se consigue establecer una línea de comunicación entre las subredes. La función de la mula de datos sería la de descargar (y cargar) la información necesaria, al encontrase situada cerca de un nodo perteneciente a una de las subredes.

Existen diferentes alternativas para actuar como mulas de datos en términos de hardware y software. Una buena opción es la tecnología SunSPOT (de la empresa Orace Corporation), utiliza el lenguaje de programación Java y su máquina virtual *Squawk*[10] para el sistema operativo.

En el caso de este proyecto, se ha optado por utilizar Arduinos, en concreto Arduinos NANO debido a su bajo coste. El lenguaje que utilizan estos microprocesadores está basado en C y C++. Un ejemplo es la empresa multinacional zaragozana Libelium, que emplea módulos llamados waspmote basados en el uso de Arduinos, aplicados a distintos ámbitos de las telecomunicaciones[20].

A continuación se van a presentar los objetivos de este proyecto.

3.2. Objetivos del proyecto

Los objetivos del presente Trabajo de Fin de Grado se encuentran resumidos en los siguientes puntos:

1. Entender y analizar las redes de sensores inalámbricos, así como las consecuencias de utilizar mulas de datos.

   Este primer objetivo es fundamental a la hora de desarrollar un sistema eficiente para aplicar a redes de sensores inalámbricos. Se tratará de analizar las ventajas e inconvenientes de emplear mulas de datos para comunicar redes situadas a grandes distancias entre si.

2. Diseñar e implementar un sistema de mula de datos funcional.

   Este objetivo abarca el diseño del software necesario para hacer el sistema de mula de datos funcional. Se debe buscar satisfacer todos los requisitos del sistema, así como mantener el nivel de consumo energético al mínimo posible, además de maximizar la eficiencia a la hora de transferir los datos.

   En cuanto a la implementación, se trata de la construcción de un prototipo que utilice el software diseñado. Se emplearán tres módulos Arduino NANO. Dos tendrán el papel de subred y un tercero actuará de nodo móvil.

3. Comprobación empírica de la funcionalidad y el rendimiento.

   Una vez diseñado e implementado el sistema de mula de datos, este objetivo conllevará una fase de pruebas para determinar la eficiencia del sistema. También se comprobará la velocidad de transmisión. Y se tratará de establecer una conclusión determinista respecto al sistema.

4. Estado del Arte

En este capítulo se estudiará la situación actual de las redes de sensores inalámbricos dispersos, incluyendo la problemática que rodea al tema y su uso como solución. Además se expondrán proyectos reales que emplean la tecnología de las mulas de datos como herramienta principal.

4.1. Redes inalámbricas dispersas

Tipicamente, las redes de sensores inalámbricos, también llamadas WSN (Wireless Sensor Network) basan su comunicación en el uso de dos arquitecturas diferentes, como se aprecia en la Figura 2:

- **Redes peer-to-peer**: se trata de redes multisalto en las que los nodos pueden transmitir la información entre ellos, sin la necesidad de un nodo central. Finalmente, la información alcanza un nodo pasarela, encargado de transmitir esa información a otras redes.

- **Redes de estrella**: están basadas en el uso de un nodo central, que se encarga de coordinar las comunicaciones de la red. Todos los nodos de la red están directamente conectados con este nodo coordinador.

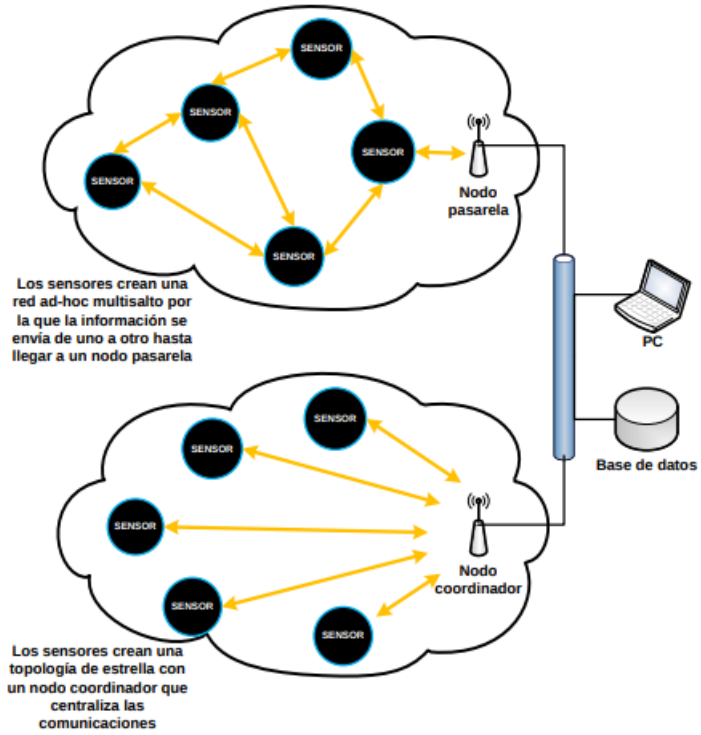

Figura 2: Arquitecturas de redes WSNs más comúnes.[30]

Las redes inalámbricas se emplean constantemente en un gran abanico de situaciones. Ambas estructuras (peer-to-peer y de estrella) pueden tener una distribución homogénea o heterogénea. En aplicaciones industriales o urbanas es posible controlar las distancias entre los nodos y desplegar redes que cubran todo el área[32]. Sin embargo, en ocasiones se necesita desplegar redes en entornos más irregulares, como puede ser una reserva natural[5], no se pueden controlar las distancias ni las condiciones del terreno entre los nodos y, a menudo, la conectividad radio entre los sensores no esta garantizada.

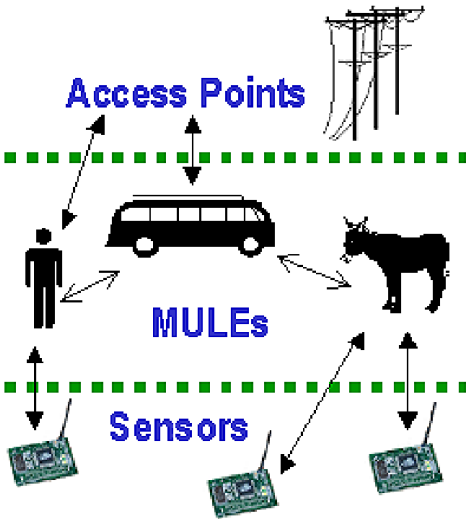

Figura 3: Representación de las comunicaciones entre los tres tipos de nodos.[27]

Cuando la situación requiere que los nodos de la red esten situados a grandes distancias se habla de SWSN (Sparse Wireless Sensor Networks) o redes de sensores dispersos[27]. Entre las soluciones al problema de la irregularidad del terreno destacan tres:

- **Instalación de nodos base**. Es la solución más directa, pero también la más cara. Consiste en instalar las suficientes estaciones base como para cubrir toda la zona de medición. Así los sensores pueden mandar directamente la información a la estación base más próxima. Adicionalmente, para aumentar el radio de comunicación emplean altas potencias de transmisión, aumentando el gasto energético.

- **Uso de redes ad-hoc**. Se basa en el uso del primer tipo de estructuras expuesto previamente, las redes peer-to-peer. Desplegando una gran densidad de nodos se hace posible que se comuniquen entre ellos, haciendo llegar la información al nodo pasarela a través de una ruta en la red ad-hoc.

- **Uso de mulas de datos**. Es la solución en la que se centra este proyecto. Consiste en el uso de nodos móviles denominados MULEs (Mobile Ubiquitous LAN Extensions [29]) o mulas de datos. La principal funcionalidad de

estos nodos es la movilidad. Cuando se acercan a los sensores de las redes estáticas pueden realizar comunicaciones de corto alcance y, de la misma forma, descargan la información recolectada al acercarse a un nodo base.

	Latencia	Potencia de transmisión	Exito de envíos	Coste de infraestructura
Estaciones Base	Baja	Alta	Alto	Alto
Redes ad-hoc	Media	Media-Baja	Medio	Medio-Alto
Mulas de datos	Alta	Baja	Medio	Bajo

Tabla 1: Comparativa del rendimiento de las tres alternativas para las WSN[27].

Cada una de estas alternativas conlleva un gasto económico diferente: desplegar una infraestructura de nodos bases tiene altos costes materiales; pero al estar la red establecida de una forma más estática se minimiza la latencia, a costa de usar una potencia de transmisión elevada.

Por otro lado, las redes ad-hoc reducen el coste de infraestructura, pero aumenta al necesitar un elevado número de nodos. En contraposición, permiten el uso de potencias de transmisión menores.

Por último, las mulas de datos. Es la alternativa con el coste más bajo, pues las potencias de transmisión son bajas y el número de nodos no es muy grande. No obstante, encuentra otros inconvenientes: como la elevada latencia que experimenta la información, pues la red no esta establecida como en el caso de las estaciones base. En el siguiente apartado se profundiza más en las mulas de datos.

4.2. El papel de las mulas de datos en las redes de sensores inalámbricos dispersos

Existen diferentes alternativas y estrategias que proponen una solucion al problema de las redes de sensores inalámbricos dispersos; durante este apartado se va a analizar que lugar ocupa el uso de nudos móviles o *data mulling* [7].

En 2003 investigadores de Intel Corporation[1] propusieron una arquitectura a tres niveles, siendo los primeros en utilizar el término MULE[27]. Se trata de una arquitectura generalista; en el diagrama de la Figura 4 se aprecian los tres niveles, correspondiendo cada uno de éstos a un tipo de nodo.

- **Nodo sensor**. Es el encargado de producir el input para la red. Su función es recoger datos físicos del entorno y almacenar estos datos hasta poder transmitirlos a un nodo mula que se encuentre a su alcance.

[1]https://www.intel.es/content/www/es/es/homepage.html

- **Nodo mula**. Se trata del nodo móvil. Es el encargado de descargar y almacenar la información que los nodos sensores han recopilado con la finalidad de transportar los datos hasta un nodo servidor cuando esté a su alcance. El nodo deberá desplazarse físicamente para entrar en el rango de transmisión de los otros dos tipos de nodos.

- **Nodo servidor**. Es el encargado de almacenar la información que el nodo mula le transmita. Además, sirve de puente para el siguiente uso de los datos: ya sea enviarlos a un servidor, procesarlos o simplemente almacenarlos para otro uso futuro.

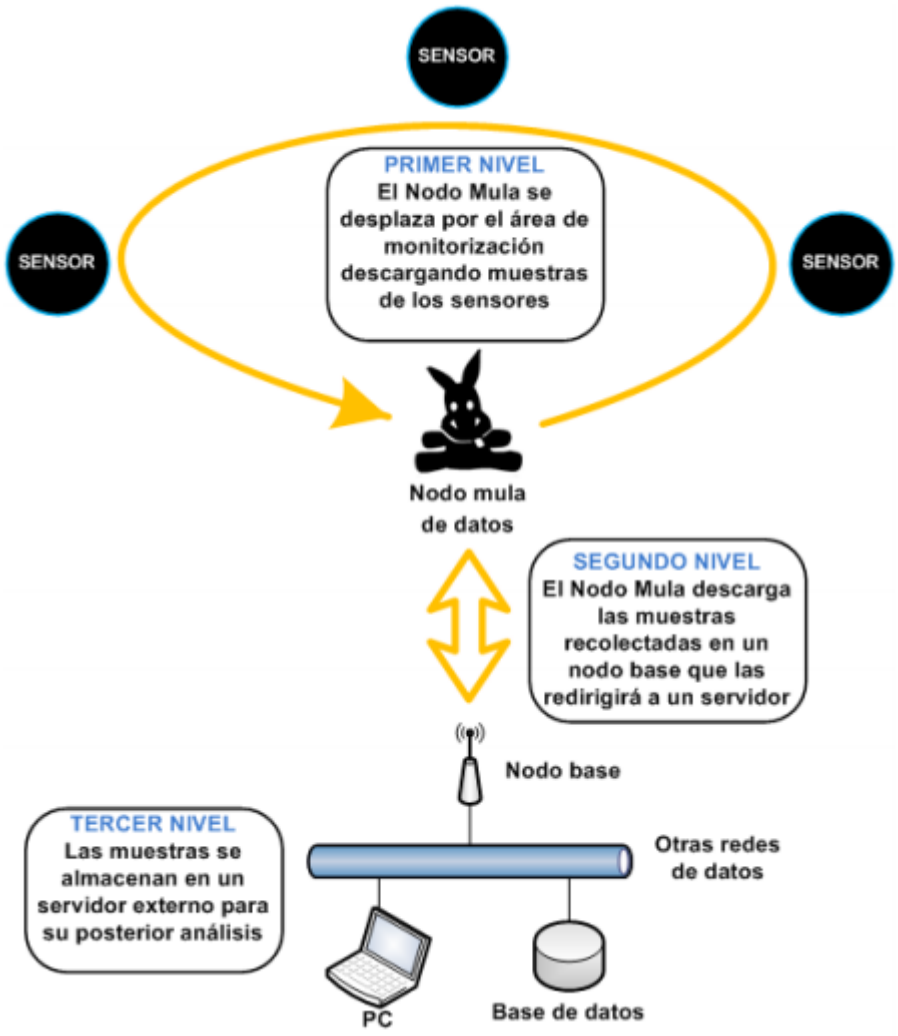

Figura 4: Esquema de tres niveles para mulas de datos.[27]

Analizando los niveles correspondientes a estos tres tipos de nodos tenemos, para comenzar, el primer nivel. Éste se basa en los nodos sensores. Está compuesto por un conjunto de nodos dispersos por el área de monitorización sin que exista

una conexión directa entre ellos. A este nivel, los nodos deben estar escuchando la llegada de una posible mula de datos que pase cerca y así poder subir las mediciones a dicha mula. Se debe de estar buscando la conexión con una mula constantemente, con el fin de reducir la latencia de la transmisión.

Continuando con el segundo nivel, éste actúa como puente entre el primer y el tercer nivel. El nodo correspondiente es el nodo mula que debe realizar dos comprobaciones de conexión con los otros dos niveles constantemente. Por una parte, debe solicitar la descarga desde los nodos sensores y, por otra, el envío de todos los datos almacenados a los nodos servidor. Es posible también que se llene el espacio de memoria del nodo mula, en cuyo caso se puede actuar de diferentes formas: no descargar información nueva o actualizar con las mediciones más actuales. Pero, en cualquier caso, sería conveniente buscar un nodo servidor lo antes posible para volver a liberar esta memoria.

Por último, el tercer nivel. Tenemos los diferentes nodos servidores que son los encargados de contestar a las peticiones de descarga del nodo mula y ofrecer la información para el siguiente paso, generalmente, almacenarla de forma persistente para analizarla con posterioridad.

Por otro lado, es importante hablar de las propias limitaciones que presenta esta arquitectura. La mayor de éstas es la latencia de transmisión de la información, dado que tanto los sensores como los servidores, dependen de que pase cerca una mula. Además, la ruta física que toma la mula no está definida, lo que hace impredecible en que momento se realizará la descarga en la misma. En estos casos se utiliza un modelo probabilístico adecuado a las características del nodo mula[27, 16, 31, 14]. Sin embargo, el estudio de dicho modelo, queda fuera del ámbito de este proyecto.

Por último, otro problema que plantea el uso de nodos sensores es el consumo energético. A pesar de no utilizar altas potencias de transmisión, deben estar escuchando constantemente la posible llegada de un nodo mula. Puede ser un inconveniente a la hora de tener que recargar estos nodos sensores. Ya existen estudios que profundizan en este problema [4, 14].

4.3. Ejemplos de uso de mulas de datos

A continuación se presentan tres ejemplos de proyectos reales en los que se ha optado por sistemas de mulas de datos.

4.3.1. Proyecto ZebraNet

El proyecto ZebraNet[2] fue desarrollado por la Universidad de Princeton de Estados Unidos en 2002. Su objetivo era asistir en el estudio de la fauna de la reserva natural Sweetwaters de Kenia para estudiar las migraciones de las cebras[17].

El proyecto está basado en colocar nodos sensores colocados en collares diseñados especialmente para las cebras, que recopilan la posición GPS (Global Position System) de su portador. Las cebras migran y se mueven con libertad por la reserva natural, creando una red de sensores dispersos que además se encuentra en movimiento.

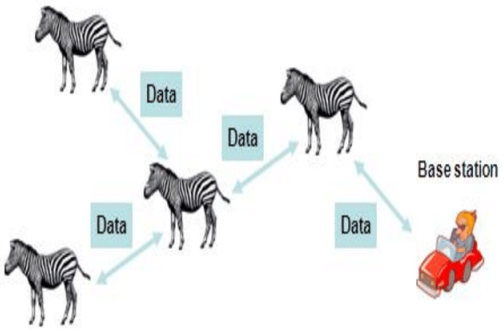

Figura 5: Representación de las conexiones entre cebras y la mula de datos[25].

Figura 6: Fotografía del collar de las cebras[34].

Los collares de Zebranet disponen además de la posibilidad de comunicarse entre ellos, cuando se encuentran en rango de otro nodo sensor, comparten vía radio los registros almacenados y guardan la información de todos los encuentros con otros collares. De esta forma se crea una red peer-to-peer entre los sensores de las cebras de una manada (en cierto modo, los sensores también están actuando como mulas de datos). Después, periódicamente, se recopilan los datos almacenados en los collares de las cebras mediante el uso de vehículos (ya sean terrestres o aéreos) que se acercan al área de monitorización mediante comunicaciones inalámbricas.

[2]https://www.princeton.edu/news/2002/11/12/engineers-and-biologists-design-wireless-zebranet

4.3.2. Proyecto Sensornets

Sensornets[3] es un proyecto desarrollado por la Universidad de Sur California en el año 2011. El proyecto se basa en utilizar los teléfonos móviles personales como mulas de datos, dado que existe un elevado número de dispositivos y podrían suponer una forma barata de transmitir información[23]. Podrían ser utilizados como un eficiente sistema de mula de datos por tres razones principales, explican los desarrolladores de Sensornets.

Figura 7: Representación de los tres nodos en el proyecto Sensornets[22].

En primer lugar, los teléfonos móviles están por todas partes (la ITU, International Telecommunication Union, estima que había 4.600 millones de usuarios con móviles en 2011) lo que supone que alrededor del 70 % de las personas ya llevan un móvil encima. Además, los teléfonos son más comunes en zonas con altas densidades de población, donde las mulas de datos dispondrían de más posibles comunicaciones. La segunda razón está ligada a esta primera, y es que la movilidad de las mulas de datos estaría cubierta, dado que los usuarios de los teléfonos móviles ayudarían a desplazar los dispositivos.

Por último, la tercera razón es el hecho de que los móviles modernos tienen un poder computacional elevado. Y, sin la necesidad de desarrollar e investigar específicamente para las mulas de datos, los dispositivos móviles tienen muchas funciones que se están desarrollando para los móviles en sí, que las mulas de datos podrían aprovechar; como comunicaciones de corto alcance (Bluetooth), a largo alcance (3G / 4G), funciones de ahorro de energía, etc.

[3] ftp://ftp.mosis.org/isi-pubs/tr-673.pdf

4.3.3. Proyecto DakNet

El proyecto DakNet[4] fue desarrollado por investigadores del MIT (Massachusetts Institute of Technology) en el año 2003 para llevar las redes de datos a pequeñas poblaciones rurales de países en vías de desarrollo. En concreto, los países en los que se han desplegado redes de DakNet son India y Camboya[24].

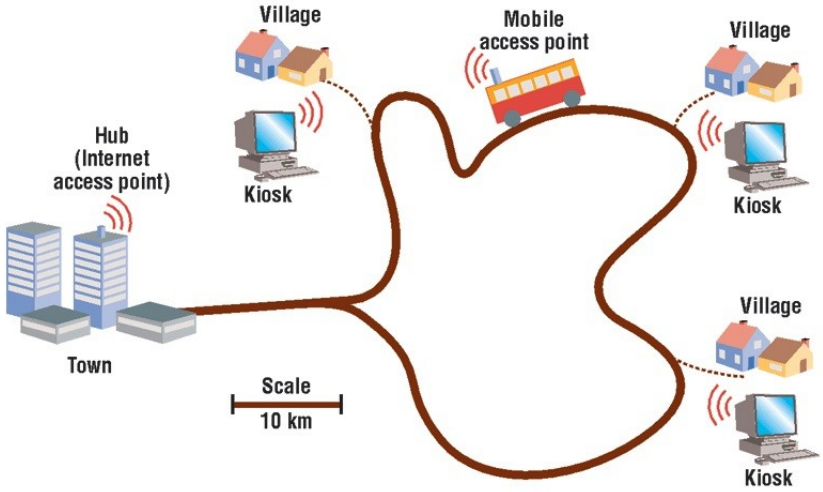

Figura 8: Arquitectura básica del proyecto DakNet.[28]

La arquitectura (apreciable en la figura 8) se basa en una serie de puntos de acceso públicos denominados "kioscos", éstos ofrecen conectividad a través del uso de mulas de datos denominadas MAPs (Mobile Access Points), dado que son mulas de datos, la conexión no se produce a tiempo real. El proyecto utiliza como nodos móviles autobuses públicos o el vehículo del cartero local. Los MAPs actúan descargando la información de los kioskos mediante conexiones de baja frecuencia, para después transportarla hasta un punto que sí tenga acceso a Internet a tiempo real.

Así, se pueden llevar servicios telemáticos a zonas donde, por diferentes razones, no resulta conveniente llevar una infraestructura tradicional de comunicaciones. Sin embargo, hay que tener en cuenta el hecho de que la conexión no se produce a tiempo real, por lo que muchos servicios deberán ser adaptados[18, 13].

[4]https://www.media.mit.edu/publications/daknet-rethinking-connectivity-in-developing-nations/

4.4. Conclusiones del capítulo

Durante las secciones anteriores se han expuesto tres proyectos que plantean la utilización de mulas de datos así como el marco tecnológico que las rodea. Cada uno de los proyectos presenta un marco diferente y justifica su uso por una variedad de razones.

Por una parte, el proyecto ZebraNet se desarrolla en un escenario en el que no tiene sentido desplegar una infraestructura de telecomunicaciones pues se basa en la movilidad de los nodos sensores. En contraposición, el proyecto de Sensornets plantea el uso de las mulas de datos por razones completamente opuestas a ZebraNet o DakNet: existe un gran nivel de infraestructura y resulta conviente aprovecharla al máximo.

Por último, DakNet plantea una alternativa económica a desplegar una infraestructura de telecomunicaciones. Además, el sistema DakNet se puede implementar y poner en funcionamiento considerablemente más rápido que otros sistemas tradicionales.

En conclusión, en cualquier situación se puede considerar y estudiar la posible implementación de una mula de datos, dado que a su movilidad innata presenta una versatilidad que otros sistemas de comunicaciones inalámbricas no tienen. Sin embargo, a la hora de plantearse el uso de un sistema de mula de datos se deben tener en cuenta las desventajas que éstos presentan.

5. Tecnologías utilizadas

Como se ha mencionado con anterioridad en este proyecto se van a utilizar módulos Arduino. En concreto, tres módulos de Arduino NANO. En cuanto a la comunicación, se va a hacer uso de módulos RF 433MHz. Se trata de módulos muy baratos que cuentan con un par emisor (FS1000A) receptor (XY-MK-5V)[19]. Además, se utilizarán otros elementos electrónicos básicos, como resistencias o LED's. En los siguientes apartados se profundiza más en las características de estos elementos.

5.1. Arduino

Arduino es una plataforma open-source fácil de usar tanto en términos de hardware como de software. Las placas Arduino leen información a través de sensores, botones o incluso una señal online. Procesan estos inputs y los transforman en outputs, activando diferentes actuadores como motores, LED's, o enviando un mensaje. El cómo transforman la información que obtienen es determinado mediante un set de instrucciones que el usuario puede enviar al microprocesador de la placa Arduino a través de un conector USB mini-B.

A diferencia de otros microprocesadores, Arduino tiene una relación directa con la electrónica que rodea al dispositivo. Para hacer una interacción con elementos del sistema (LEDs, Botones, Antenas, Motores...) es necesario diseñar y construir un circuito electrónico alrededor de la placa Arduino. Dicha interacción se realiza a través de pines conectados con el microcontrolador, que es el componente donde se carga el código escrito. De esta forma se aumenta significativamente las posibilidades que un dispositivo Arduino ofrece, pues se eliminan las limitaciones de hardware al permitir conectar cualquier sistema electrónico a la placa. No obstante, cabe destacar que la potencia proporcionada por una placa Arduino es limitada, así, para circuitos electrónicos de tamaño medio o grande se deberá recurrir a una fuente alternativa.

5.1.1. Características del dispositivo Arduino NANO

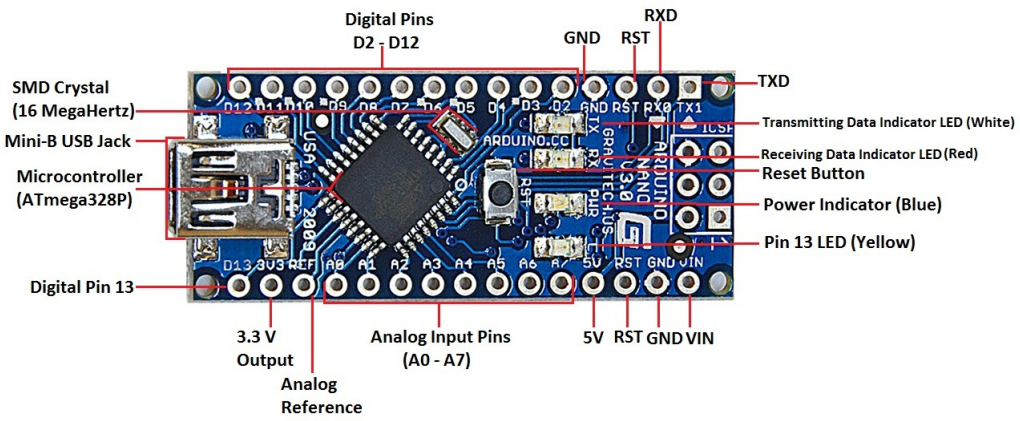

Figura 9: Esquema de un Arduino NANO.[9]

En el caso concreto de este proyecto, se han utilizado placas Arduino NANO (Figura 9), son unas placas que cuentan con un microprocesador ATMega328P. Se trata de un microprocesador de 8 bits con un consumo de energía bajo, diseñados para optimizar el ratio entre consumo y velocidad de procesamiento[15]. También tienen integrada una memoria flash de 32KB, con 2KB reservados para el gestor de arranque (bootloader). Como casi todas las placas Arduino, el valor del voltaje para la lógica binaria es $5V$. Se trata de unas placas compactas por lo que sólo tienen 22 pines digitales (que pueden actuar como input o como output) y 8 pines analógicos[26] (que sólo son utilizados como input, su uso más común es leer la información que proporcionan los sensores).

Estas placas cuentan además con la opción de programación ISP (In-System-Programmer), utilizada para programar microcontroladores AVR (microcrontoladores de la compañia Atmel, cómo es el caso de los Arduino NANO). La programación ISP es un método de programación a través de otra placa Arduino [6]. Es posible programar el microprocesador de una placa (placa objetivo) desde una placa conectada al ordenador (placa programadora) mediante un set de pines dedicados a este propósito. En la Figura 9, se pueden apreciar estos 6 pines a la derecha en la placa. Optar por la opción de programar a través de los conectores ISP permite eliminar el gestor de arranque (bootloader) de la placa objetivo, liberando la parte de la memoria que normalmente queda reservada para éste. Sin embargo, en la mula de datos de este proyecto no debería ser necesario necesitar más memoria que los 30KB de los que disponen las placas Arduino NANO después de reservar el espacio para el gestor de arranque.

En cuanto al aspecto económico de los dispositivos Arduino NANO, el precio de las placas es muy dependiente del fabricante, por lo que éste varía en un rango considerable. Sin embargo, dado el carácter abierto del software y del hardware relativo a Arduino, existen muchos fabricantes que ofrecen alternativas más

económicas. Este rango de precios es aplicable a todas las placas Arduino y no sólo a las NANO, no obstante, en este proyecto se ha optado por las placas NANO.

5.2. Lenguaje de programación Arduino

El lenguaje de programación que utilizan las placas de Arduino está basado en C++. Cuenta con una librería propia y muchas de las funciones están diseñadas específicamente para los dispositivos Arduino. Sin embargo, es posible utilizar funciones de C++. C++, a su vez, se trata de una ampliación de C para hacer el lenguaje más cercano a la programación orientada a objetos[2].

Por otra parte, la mayoría de los Arduino tienen un microcontrolador AVR fabricado por Atmel[5] lo que hace posible la programación en otros lenguajes. Generalmente, se utiliza una adaptación de C++ que proviene de avr-libc, avr-gcc, y avr-binutils; librerías que a su vez están basadas en librerías de C con GCC, GNU Compiler Collection.

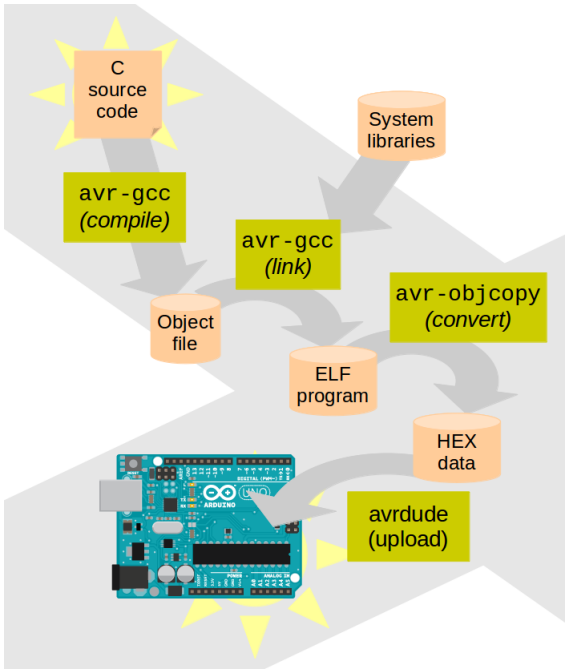

Figura 10: Proceso de compilación y subida a una placa Arduino.[3]

GCC es un conjunto de compiladores estandarizado para los Sistemas Operativos derivados de UNIX. Cuando GCC se emplea para generar código para un microcontrolador AVR, se denomina avr-gcc. Éste es el compilador que utiliza el IDE, Integrated Development Environment, de Arduino (el entorno de desarollo más común que se verá en más profundidad en la sección 5.2.1).

[5]https://start.atmel.com/

Para programar una placa Arduino no es necesario conocer todo lo mencionado anteriormente. Una de las mayores ventajas de Arduino es la sencillez del lenguaje. Éste ofrece librerías que facilitan la programación de los pines de entrada, de salida y los puertos de comunicación así como otras librerías más específicas. De hecho, es uno de los hardware más accesibles para los usuarios por lo que existen numerosos proyectos que emplean scripts publicados en Arduino y de los que es posible aprender y adaptar a otros proyectos.

Una de las mayores diferencias que el modo de ejecución del lenguaje Arduino tiene con un código estándar de C++ es que, de forma predeterminada, el código principal compilado y subido a la placa Arduino se ejecuta en bucle de forma infinita. En el caso de este proyecto esta ejecución en bucle es conveniente (como se expone en más detalle en la sección 6) pues, los nodos sensores y servidores lanzan peticiones de forma repetida a los nodos mula sin necesidad de complicar el código. Además, hace posible el uso de una estructura del código muy modular, simplificando la interpretación del código.

5.2.1. Entorno de desarrollo

El software empleado durante este proyecto es el IDE (Integrated Development Environment) de Arduino, en concreto, se ha utilizado la versión 1.8.4[6] del software. Se trata de una herramienta de código abierto (Open Source) desarrollada específicamente para las placas Arduino, incluye un amplio abanico de librerías y drivers utilizados por dichas placas.

El IDE de Arduino incluye el paquete de compiladores GCC mencionado en la sección 5.2, creando un puente directo entre el lenguaje de programación a alto nivel, programado por el usuario en el editor de texto que el IDE ofrece, y el bajo nivel, interpretado por el microprocesador de la placa Arduino. Adicionalmente, es relevante mencionar que usualmente se llama sketches al código escrito en este editor.

Además de las funciones comunes de un editor de texto, el IDE de Arduino también cuenta con opciones de configuración directa para las placas Arduino. El software contiene los drivers necesarios para utilizar todas las placas Arduino más comunes: UNO, MEGA, NANO, etc. Facilita la comunicación a través de los puertos USB(también llamados puertos Serial) con los microprocesadores, así como una pantalla de monitorización del Serial que ofrece la posibilidad de comunicarse con la placa en ambas direcciones. La pantalla de monitorización del Serial es especialmente útil para depurar (*debuggear*) los sketches, pues permite mostrar por pantalla mensajes de forma directa.

En definitiva, es recomendable que cualquier sistema que utilice hardware de Arduino en cualquiera de sus partes utilice el software de Arduino desarrollado específicamente para dicho hardware, pues simplifica tanto el código como el proceso

[6]https://www.arduino.cc/

de programación.

5.2.2. Librería de terceros: Virtual Wire

La librería Virtual Wire[7] es una librería de comunicaciones desarrollada por Mike McCauley que permite la comunicación de múltiples Arduinos utilizando transmisores baratos de RadioFrecuencia (RF)[1].

Virtual Wire cuenta con herramientas para enviar mensajes cortos sin contestación. Es capaz de soportar una cantidad moderada de transmisores y receptores de radio. Cuenta con un número reducido de funciones dedicadas específicamente a la configuración del microprocesador, al envío desde el transmisor, y la recepción desde el receptor.

En cuanto a la parte de configuración, es necesario indicar que pines del Arduino serán dedicados al transporte de datos desde/hasta el microprocesador. El núcleo de la comunicación se basa en dos funciones:

- **vw_send(message, length)** Es la función encargada de enviar el mensaje. Sus argumentos son la posición del primer byte de dicho mensaje así como la longitud del mismo.

- **vw_get_message(buf, &buflen)** Es la función de recepción estándar. Similar a la función de envío, necesita un array donde almacenar la información y el tamaño de dicho array, que se asigna según el tamaño del mensaje recibido.

Adicionalmente, existen diferentes funciones para activar y desactivar el proceso de recepción, así como realizar esperas hasta obtener un mensaje enviado por el transmisor.

5.3. Hardware adicional del proyecto

Como se ha mencionado anteriormente los módulos de Arduino tienen una relación directa con la electrónica, dado que hay que diseñar todos los circuitos que van más allá del microprocesador.

En la situación del presente proyecto se han utilizado circuitos simples para encender LED's, que actúan como indicadores de diferente información; botones, utilizados para comunicar información al Arduino; y los módulos de radio frecuencia para las telecomunicaciones.

[7] https://www.pjrc.com/teensy/td_libs_VirtualWire.html

5.3.1. Módulos RF: Emisor y Receptor

Existen diferentes alternativas para realizar comunicaciones inalámbricas entre Arduinos. En este proyecto se ha optado por utilizar unos módulos simples que utilizan el espectro de radiofrecuencia como herramienta principal. Se basan en la generación de ondas electromagnéticas al aplicar una corriente alterna a una antena.

Específicamente, los módulos utilizados trabajan en la frecuencia de $433MHz$, por lo que se encuentran en un rango medio dentro del espectro[19]. Esta frecuencia pertenece a una banda libre, por lo que su uso es gratuito. Los módulos de recepción y emisión se llaman XY-MK-5V y FS1000A respectivamente, y son populares por su bajo coste. En las figuras 11 y 12 se pueden ver las conexiones de estos módulos con una placa Arduino NANO.

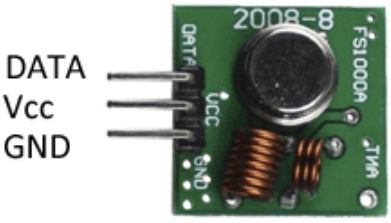

Figura 11: Módulo XY-MK-5V[19]

Figura 12: Módulo FS1000A[19].

El alcance de los módulos dependen principalmente del voltaje que alimente la antena. Con el voltaje que proporciona la placa Arduino ($5V$) y la antena integrada en el módulo el rango es escaso, de apróximadamente 2 ó 3 metros; mientras que con una mayor antena y $12V$ el módulo puede alcanzar 300 metros.

La comunicación es unidireccional, de canal único y tiene una baja velocidad de transmisión. Utiliza la modulación ASK (Amplitude Shift Keying) para realizar la comunicación. Consiste en un tipo de modulación de la amplitud que codifica los datos binarios en variaciones de la amplitud de una señal. Adicionalmente, los dispositivos no disponen de filtro ni de un identificador propio del hardware, por lo que para para hacer la comunicación más robusta se deberá implementar a través del software.

En definitiva, los módulos RF son una alternativa barata para proyectos a pequeña y media escala aunque hay que tener presente el hecho de que los módulos en si mismos realizan una comunicación simple y, con el fin de mejorar la fiabilidad de la comunicación, el software se debe desarrollar en consecuencia.

6. Diseño e Implementación

6.1. Introducción

Durante esta sección se pretende describir el diseño de la plataforma software propuesta para este proyecto, así como el montaje del hardware. El primer paso es analizar las funciones y requisitos que el sistema debe satisfacer.

Se profundizará en el diseño del circuito electrónico de los distintos componentes del sistema (nodo sensor, mula y servidor), la estructura estática de cada uno de ellos, el protocolo de comunicación (la arquitectura del mensaje), y la implementación del conjunto.

Así, se pretende crear una idea del funcionamiento del conjunto de la plataforma, además de crear una referencia para la implementación del prototipo.

6.2. Requisitos del sistema

El modo primario de actuación del sistema es el Modo Data Mulling mediante el uso de Arduinos. Para ello se va a seguir el esquema de arquitectura a tres niveles mencionado en el apartado 4.2 y diseñada por Intel Corporation [27]. La principal característica de dicha arquitectura es que tantos los nodos sensores no se comunican directamente con los nodos servidores y siempre que se comunican es a través de un nodo mula intermediario. Además, los nodos sensores tampoco se comunican entre ellos, creando un esquema similar a la red de estrella con el nodo mula.

Los nodos sensores realizan mediciones periódicas del área de monitorización y las almacena de forma persistente hasta la llegada de un nodo mula que las descargue y permita el vaciado de la memoria del nodo sensor. Cuando un nodo mula pasa cerca de un nodo sensor se produce dicha descarga. Después, el nodo mula se desplaza físicamente hasta una ubicación cercana a un nodo servidor y éste descarga todas las muestras que fueron originalmente recolectadas por los nodos sensores.

Teniendo en mente este modo de funcionamiento, a continuación se definen los requisitos primarios que la plataforma diseñada debe cumplir:

- **Nodo sensor.**
    1. Muestrear de forma periódica y en función de la configuración proporcionada diferentes magnitudes físicas. En el caso de una planta industrial dichas magnitudes pueden ser muy variadas: temperatura, niveles de ruido, luminancia, velocidad de una cinta, nivel de líquido en el interior de un tanque, etc. Además, es conveniente que entre las mediciones también se obtenga el valor de la batería de la carga que alimenta el propio nodo sensor.

2. Almacenar las muestras de las mediciones de forma persistente (hasta la llegada de un nodo mula).

3. Escuchar las solicitudes de descarga de un nodo mula próximo.

4. Descargar las mediciones a un nodo mula cuando éste lo solicite y la comunicación esté disponible.

5. Almacenar la configuración del nodo sensor de forma persistente hasta que se reciban nuevas instrucciones.

- **Nodo Mula.**

1. Comprobar de forma periódica la comunicación a un nodo sensor para descargar sus muestras.

2. Descargar las muestras de un nodo sensor próximo.

3. Almacenar las muestras descargadas de un nodo sensor.

4. Buscar de forma periódica a un nodo servidor próximo que se encuentre disponible.

5. Descargar los datos almacenados de los nodos sensores en un nodo servidor.

6. Almacenar la configuración del nodo mula de forma persistente hasta que se reciban nuevas instrucciones.

7. Hacer de intermediario para las órdenes enviadas desde un nodo servidor hasta un nodo sensor.

- **Nodo Servidor.**

1. Descargar la configuración y las muestras almacenadas de un nodo mula. Se podría descargar también de un nodo sensor en el caso de no seguir la arquitectura de tres niveles de Intel.

2. Almacenar las muestras descargadas de forma persistente.

3. Configurar en un nodo sensor las magnitudes a medir, incluyendo los períodos de muestreo y los umbrales de decisión. Con la posibilidad de hacer dicha configuración a través de un nodo mula.

4. Configurar en un nodo mula el período de búsqueda de un nodo sensor de un nodo servidor.

5. Borrar en un nodo sensor o mula la memoria persistente.

Las funcionalidades descritas anteriormente son la parte básica para llevar a cabo la plataforma propuesta. Sin embargo, existen una serie de requisitos secundarios que favorecen la funcionalidad del sistema. Algunas de estas funcionalidades son:

- **Requisitos secundarios.**
  - En general, reducir el consumo energético de todos los nodos: pero especialmente en los nodos sensores pues sus baterías son las más complicadas de reemplazar.
  - Configurar posibles alarmas que se activen cuando una magnitud física medida alcance un valor límite (configurado previamente). Esta funcionalidad debe ser implementada en los tres niveles: el nodo sensor debe detectar y lanzar la alarma, y el nodo servidor deberá mandar algún tipo de aviso al usuario.
  - En cuanto al nodo servidor, implementar la gestión gráfica de la base de datos, para facilitar la visualización de las mediciones. Se puede realizar mediante la utilización de tablas que permitan la eliminación y exportación de los datos almacenados.

Tanto los requisitos primarios como los secundarios se pueden resumir de forma genérica en nueve casos de uso, divididos a su vez en dos grupos: el primer grupo corresponde a los casos de uso de los nodos sensores y mula (figura 13), y el segundo grupo es el asociado a los casos de uso del usuario y el nodo servidor (figura 14). Existen casos de uso que sirven para más de un actor simultáneamente.

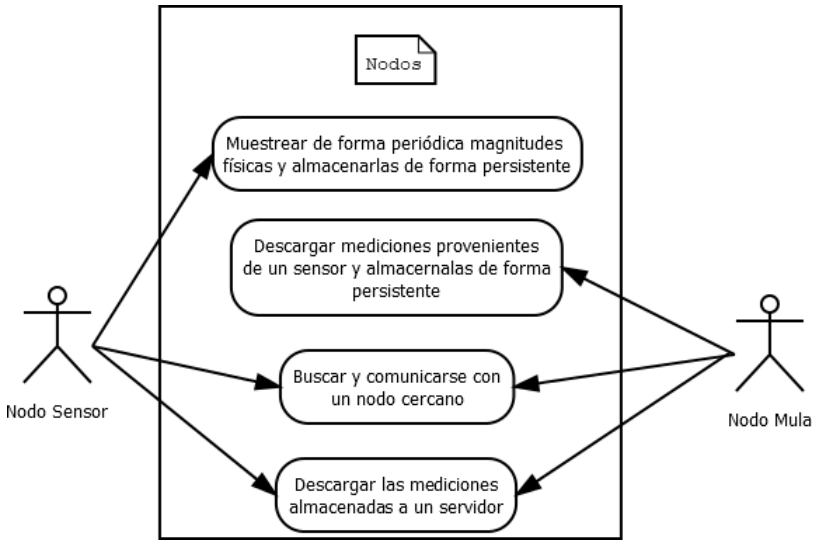

Figura 13: Casos de uso de los nodos tipo sensor y tipo mula.

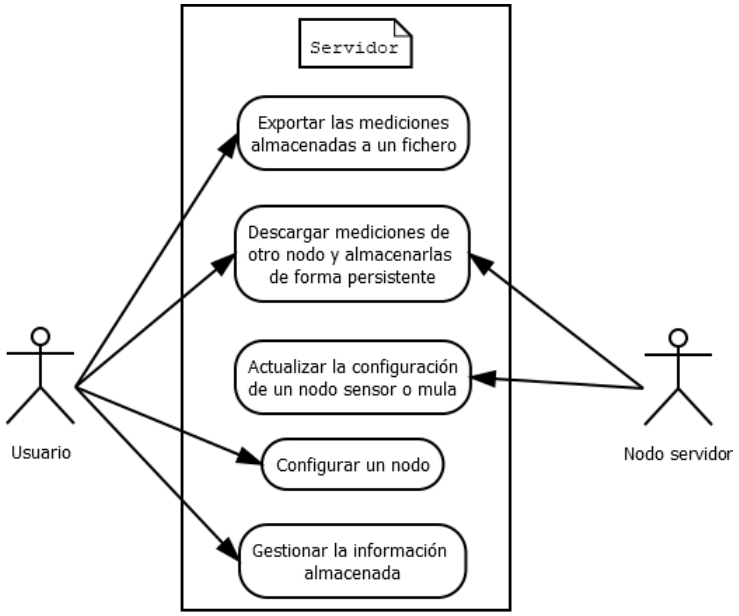

Figura 14: Casos de uso del nodo servidor y del usuario.

6.3. Circuitos electrónicos y Estructuras estáticas

6.3.1. Electrónica común a los tres nodos

En primer lugar y dado que los módulos RF tienen una comunicación de tipo unidireccional, todos los nodos han de tener tanto un emisor (o transmisor) como un receptor para poder realizar comunicaciones en ambos sentidos.

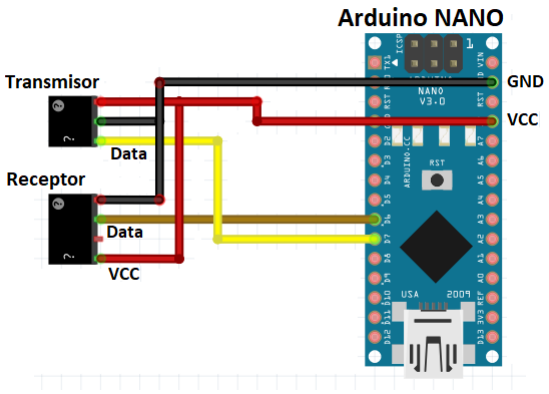

Figura 15: Esquema de las conexiones del transmisor y receptor RF.

Además de la alimentación de los módulos RF (V_{cc} y GND) se debe configurar uno de los pines digitales del Arduino para la transmisión de datos, uno para el emisor y otro para el receptor; en este caso se ha utilizado el pin D6 para el emisor y el pin D7 para el receptor. Así, los tres tipos de nodos utilizan el emisor y el receptor conectados como se aprecia en la figura 15 Por otra parte, al haber utilizado los módulos XY-MK-5V y FS1000A no es necesario emplear ningún tipo de elemento básico adicional (resitencias, diodos...) pues estos dos módulos en concreto tienen la circuitería necesaria incluida en sus PCB (Printed Circuit Board, placa de circuito impreso).

6.3.2. Nodo sensor

El requisito primario que el nodo sensor debe cumplir es el de realizar mediciones periódicas cada cierto tiempo, a la frecuencia de muestreo indicada por la configuración. Para ello debe tener componentes sensores que realicen una lectura y el Arduino obtenga dicha lectura a través de un pin analógico configurado para este propósito.

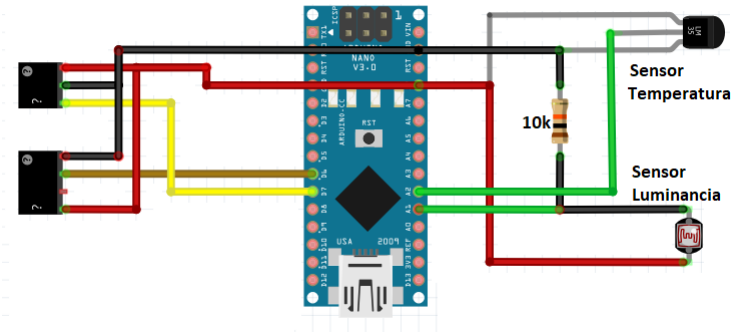

Figura 16: Circuito electrónico para el nodo sensor con sensor de temperatura y de luminancia.

Dependiendo del tipo de sensor la circuitería cambia ligeramente. En la Figura 20, se puede observar como el sensor de luminancia necesita una resistencia asociada de $10k\Omega$ para crear una corriente (y por tanto un voltaje) en el punto de lectura y, sin embargo, el sensor de temperatura está fabricado de tal forma que dicha resistencia no es necesaria.

Además, en cuanto al nodo sensor es conveniente no añadir demasiados elementos de hardware que consuman energía, pues uno de sus requisitos secundarios es el maximizar la duración de las baterías, por lo que es conveniente realizar avisos utilizando el nodo mula en lugar de indicaciones visuales (como un LED encendido durante horas).

Pasando a la parte correspondiente al software, la estructura estática, la figura 17 muestra los diferentes gestores del nodo. La multiplicidad de estos gestores es uno.

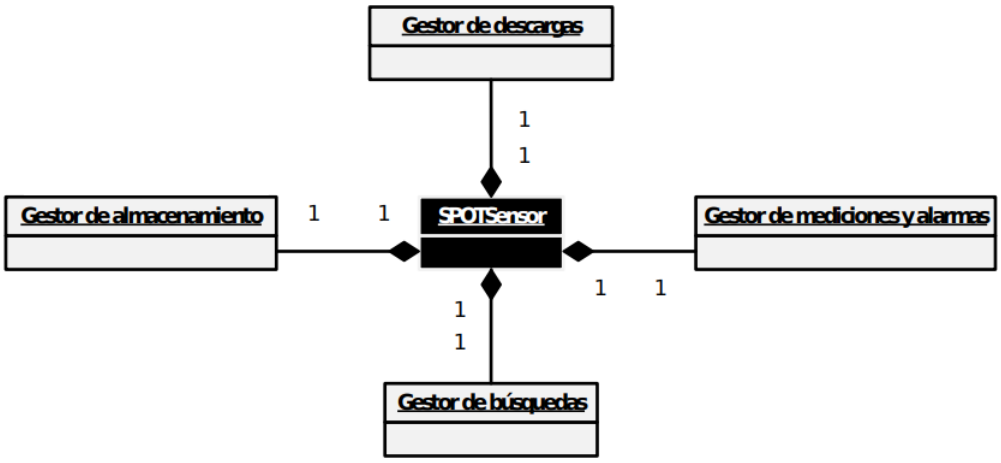

Figura 17: Estructura estática del componente sensor[8].

- El gestor de almacenamiento es el módulo encargado de guardar la información de forma persistente en el nodo sensor. Para ello, se utilizan funciones de memoria dinámica de C++. También, en caso de recibir la orden, debe realizar un borrado de la memoria para liberar espacio en la ROM.

- El gestor de búsquedas es el módulo que realiza comprobaciones periódicas para encontrar un nodo cercano disponible, ya sea un nodo mula o un nodo servidor.

- El gestor de descargas se encarga de realizar el proceso de envío de datos una vez el gestor de búsquedas ha encontrado un nodo con el que realizar la comunicación. También debe descargar la nueva configuración proveniente de otro nodo.

- El gestor de mediciones y alarmas es el módulo encargado de leer las mediciones de las magnitudes físicas monitorizadas. Adicionalmente, puede cumplir el requisito secundario de comprobación de umbrales y lanzar los consecuentes avisos.

6.3.3. Nodo mula

El nodo mula debe desplazarse de una localización a otra, y por tanto, similar al nodo sensor es conveniente no tener demasiado hardware adicional que drene la batería a mayor velocidad. Por ello, se ha optado por utilizar sólo una indicación visual a través del uso de un LED. Esta indicación se activará en dos situaciones: la batería restante es baja, o la memoria ROM de la placa Arduino está llena.

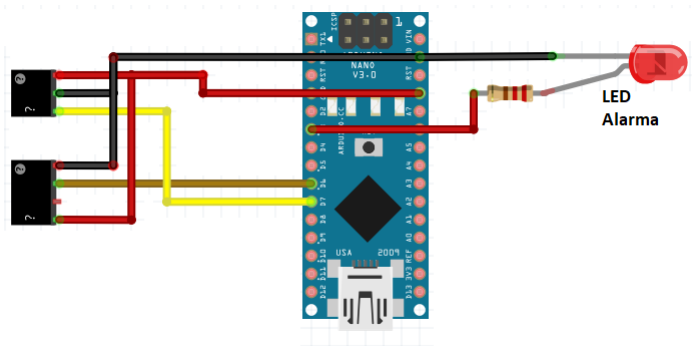

Figura 18: Circuito electrónico para el nodo mula.

En cuanto a la estructura estática del componente Mula, se presenta la figura 19 que muestra los gestores de este tipo de nodo. La multiplicidad de los gestores del componente mula es uno.

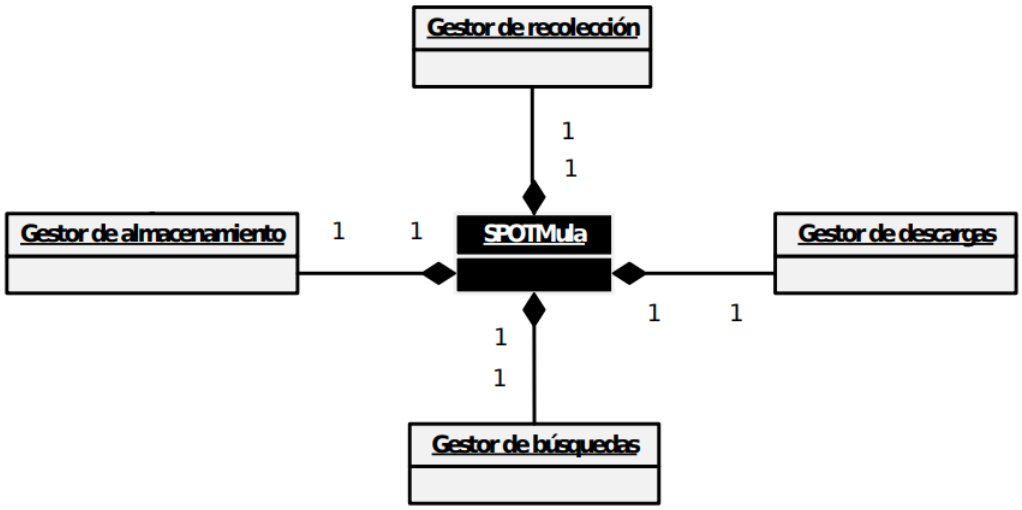

Figura 19: Estructura estática del componente mula[8].

- El gestor de almacenamiento es el módulo encargado de guardar la información de forma persistente en el nodo sensor. Para ello, se utilizan funciones de memoria dinámica de C++. Tiene el mismo funcionamiento que en el nodo sensor.

- El gestor de búsquedas es el módulo que realiza comprobaciones periódicas para encontrar un nodo servidor cercano disponible.

- El gestor de descargas se encarga de enviar las medidas almacenadas de forma persistente a un nodo servidor disponible.

- El gestor de recolección se encarga de enviar peticiones de descarga a los nodos sensores periódicamente y de recopilar la información que estos nodos tienen almacenada.

6.3.4. Nodo servidor

El nodo servidor es el puente entre el usuario y todo el sistema de mula de datos. Así, se debe tener un bus de información conectado desde un Servidor hasta el puerto Serial de la placa Arduino. A través de éste, se proporcionarán todas las instrucciones necesarias para modificar la configuración de los diferentes nodos.

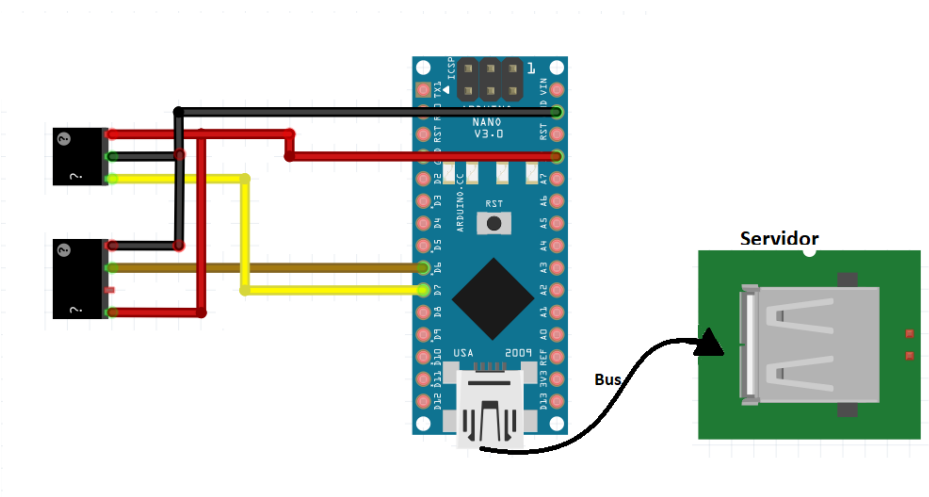

Figura 20: Circuito electrónico para el nodo servidor.

Además, la conexión del bus es bidireccional de tal forma que el Servidor también es una forma de mostrar de una forma visual la información contenida dentro de los microprocesadores de las placas Arduino.

A continuación, en la figura 21 se muestra la estructura estática del componente servidor. De nuevo, la multiplicidad de los gestores del componente es uno.

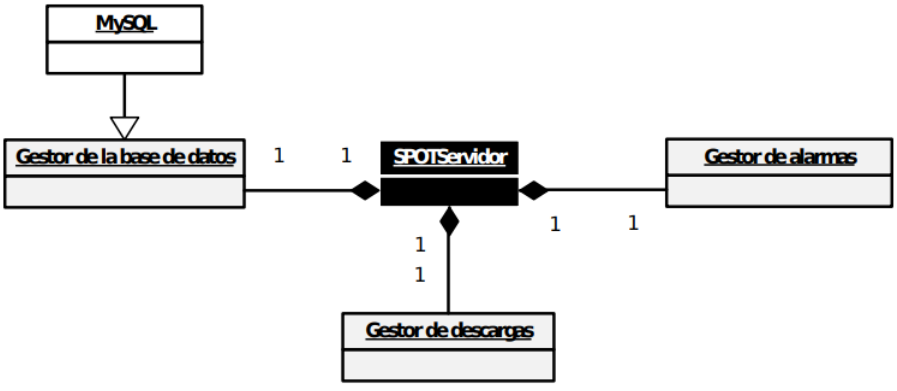

Figura 21: Estructura estática del componente servidor[8].

- El gestor de descargas es el módulo encargado de realizar la descarga de los datos contenidos tanto en los nodos mula como en los nodo sensor.

- El gestor de la base de datos se encargará de recuperar y almacenar la información de forma persistente en el Servidor. El gestor debe tener un sistema de gestión de base de datos como es MySQL[8], sin embargo, en este proyecto no se ha implementado y se comentará más en la sección 8.2.

- El gestor de alarmas forma parte de los requisitos secundarios y se encarga de escuchar las alarmas de los nodos cercanos así como de enviar el aviso correspondiente al usuario.

[8]https://www.mysql.com/

6.4. Protocolo de comunicación

Durante esta sección se van a estudiar más específicamente las especificaciones del protocolo de comunicación del sistema, es decir, la estructura del propio mensaje.

Como se ha comentado en apartados anteriores, los módulos de comunicación RF simple para Arduino no cuentan con ningún recurso además de el de enviar un mensaje. Por ello, no se puede recurrir a funciones de tipo ACK (*acknowledgement*) ya implementadas en el sistema interno del hardware. Además, los dispositivos funcionan en modo difusión (*broadcast*) de forma predeterminada y, es posible, que el sistema requiera la comunicación nivel uno a uno.

Así, con el fin de robustecer la comunicación y dar lugar a más posibilidades dentro del sistema, se ha diseñado un protocolo de comunicación basado en datagramas. Los datagramas son los bloques de información que encontramos en el flujo de información de bytes constante. Los datagramas están compuestos de dos partes: una cabecera que incluye información de control y el bloque de datos a transmitir. Utilizándolos se evita la necesidad de utilizar un gestor de secuencias que indique el principio y final de cada mensaje.

El protocolo que se ha diseñado está formado por varias capas: capa de Aplicación, capa de Seguridad, capa de Transporte y, finalmente la capa Física. La capa de Transporte no es necesaria para todos los casos, pero debido al hardware utilizado, en este proyecto si lo es.

6.4.1. Capa de Aplicación

La capa de Aplicación es la encargada de recopilar todos los parámetros del mensaje. Es la capa a la que el usuario (o la configuración establecida por el usuario) tiene acceso, de manera que dicho emisor puede especificar quién es el receptor, que tipo de datos se está enviando así como la ID de la aplicación.

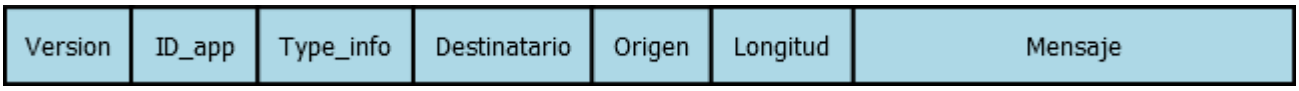

Figura 22: Estructura del mensaje en la capa de Aplicación.

La función en la que esta capa se basa tiene la siguiente declaración: [bool capa_aplicación (Mensaje, Longitud, Version, ID_app, Type_info, Destinatario, Origen);]. El objetivo de esta función es recopilar, en un único array de bytes consecutivos toda la información indicada de forma que tenga la estructura de la figura 22. A continuación se explica brevemente el significado de los argumentos:

- *Mensaje:* Es un puntero que indica la posición del primer byte del mensaje a enviar. En esta capa se trata del mensaje en si mismo, es decir, las mediciones realizadas o la configuración para el nodo sensor/mula enviada desde el servidor.

- *Longitud:* Es la longitud del mensaje, es decir, cuantos bytes ocupa el mensaje. De esta forma, el receptor conocerá número de bytes situados inmediatamente detrás del byte indicado por el puntero Mensaje.

- *Version:* Es la versión del protocolo. Actualmente es la versión 1.0, pero es posible que en el futuro se desarrollen otras versiones y se utilice este parámetro para aspectos de compatibilidad.

- *ID_app:* Es el identificador del uso de la aplicación, es posible que la tarea que está realizando el sistema cambie y con este identificador los dispositivos interpretarán que deben hacer.

- *Type_info:* Indica que tipo de datos se está enviando, es una forma conveniente de indicar que información se está enviando exactamente de cara al código de interpretación en el receptor.

- *Destinatario:* Indica, mediante una variable de tipo entero, quien es el destinatario del mensaje. Cada nodo deberá tener una ID y comprobar si este

campo coincide con dicha ID. De esta manera, se pueden realizar comunicaciones uno a uno, aunque; si el valor del campo es 0, los nodos interpretarán que el mensaje es para ellos haciendo uso de la funcionalidad broadcast.

– *Origen:* El valor está fijado según el nodo desde el que se envíe y contendrá el valor del entero identificador del nodo emisor.

Además, la función de la capa de Aplicación devolverá verdadero si la función ha conseguido realizar su tarea. Tras completar su objetivo, la capa de Aplicación accede a la capa de Seguridad indicando el inicio del nuevo array además del tamaño de éste para que la siguiente capa pueda realizar su función.

6.4.2. Capa de Seguridad

La capa de Seguridad es una capa de depuración (*debugging*). Se encarga de añadir un código único al final del array obtenido desde la capa de Aplicación, que se utilizará para identificar la validez del envío de datos.

Figura 23: Estructura del mensaje en la capa de Seguridad.

La generación del código se realiza usando una suma de verificación (*checksum*). Se trata de una función cuyo objetivo es, en un conjunto de datos, detectar si se han corrompido parte de éstos. Se realiza la comprobación antes de la emisión, se obtiene un valor que varía según el mensaje y, tras la transmisión, se realiza la misma comprobación. Si el valor obtenido en ambas comprobaciones es el mismo significa que el mensaje no ha variado y, por tanto, no se ha perdido información.

Con el fin de simplificar la suma de verificación, en este proyecto se ha optado por una función checkSum de tipo XOR. Existen funciones más complejas que realizan comprobaciones exhaustivas, como por ejemplo una función de redundancia cíclica (CRC).

A continuación la capa de Seguridad añade la fecha con el fin de mantener un registro más preciso de las telecomunicaciones, además de incluir la longitud total del array para simplificar la lectura del bloque de información en el receptor.

Similar a la capa de Aplicación, la capa de Seguridad accede a la siguiente capa: la capa de Transporte.

6.4.3. Capa de Transporte

La capa de Transporte es una tercera capa que, dependiendo del contexto donde se utilice el sistema de telecomunicaciones no tiene porque ser necesaria. Se utiliza para enviar mensajes de un tamaño grande.

En caso de que el array a enviar tuviera unas dimensiones lo suficientemente grandes como para que el hardware no fuera capaz de transmitirlo, la capa de Transporte se encargaría de desfragmentar el contenido del array en varios arrays de menor tamaño y enviar dichos arrays consecutivamente.

Cuando esto ocurre, es necesario añadir un número de secuencia que actúe como identificador del fragmento para que, a la hora de desfragmentar la información en el nodo receptor, se pueda volver a montar el array original con facilidad.

Con los diferentes fragmentos, la capa de Transporte llama a la capa Física para que ésta se encargue del paso final.

6.4.4. Capa Física

La capa Física es la más cercana a la función send de la librería Virtual Wire. Utiliza la función *vw_send(message, length)* (explicada en la sección 5.2.2) para enviar el array de datos que se ha construido en las dos capas anteriores.

Es la capa más simple pues su único objetivo es enviar el mensaje y esperar a que éste se haya enviado por completo antes de empezar otro proceso. Recibe fragmentos desde la capa de transporte que, en su conjunto, contienen toda la información a enviar. Por ello, la capa Física tiene que actuar varías veces durante el envío de un único bloque de datos.

6.4.5. Recepción del mensaje

Conociendo la estructura del mensaje que las distintas capas construyen, todos los nodos que formen parte del proceso de comunicación inalámbrica deberán tener un protocolo de recepción y desglosamiento de la información recibida.

En primer lugar, dado que todos los nodos escuchan el mensaje, es necesario comprobar la ID del destinatario y verificar si coincide con la ID del nodo que está escuchando. Si esta ID no coincide el nodo no realizará nada más. Sin embargo, si la ID coincide el nodo pasará al siguiente paso: interpretar el mensaje.

Una vez determinado que el nodo es, en efecto, el destinatario del mensaje; éste desglosará el mensaje en los distintos campos y realizará la comprobación checkSum de tipo XOR. Finalmente, si el código obtenido coincide con el que generó la capa de Seguridad, el nodo realizará la función necesaria en ese momento, ya sea almacenar la información como servidor, actualizar la configuración como sensor o transportar el mensaje como mula.

6.5. Estrategia de ahorro energético

Este último apartado describe las principales herramientas que se pueden utilizar con el fin de disminuir el consumo energético en los nodos mula y sensor, que son los dependientes de una batería. Es uno de los factores más importantes de una red de sensores inalámbricos, pues condiciona directamente la vida operativa del sistema de manera autónoma.

En primer lugar, apagar cualquier elemento de hardware que funcione de manera continuada. El ejemplo más claro son los LEDs de la placa Arduino que, a pesar de dar información relevante, consumen mucha energía al estar funcionando constantemente.

Otra medida es optimizar tanto el tiempo de muestreo como el de intervalo de búsqueda de sensores cercanos. Cuando se lanzan señales de comunicación constantemente se están activando los módulos RF aumentando el drenaje de la batería. El período de muestreo del sensor, sin embargo, está más limitado en cuanto a que es dependiente de que magnitud se este monitorizando.

En cualquier caso, en una primera instancia se escogió utilizar Arduinos NANO no sólo por su bajo coste sino porque, en comparación a otros modelos de placas Arduino (de mayor tamaño y con mayor memoria) consume poca energía. La siguiente tabla muestra el consumo de diferentes modelos:

Modelo	Consumo en mA	Duración de una batería de 1200 mAh
NANO	15	80 horas
MEGA	93	Casi 13 horas
DUE	75	16 horas
UNO	46	26 horas

Tabla 2: Tabla con el consumo energético de las placas más usadas de Arduino[11].

7. Validación del prototipo

La sección de validación del prototipo es la sección de pruebas. En ella se pretende probar tanto el hardware como el software desarrollado en la sección de diseño (sección 6). Para ello, se han realizado diferentes test que estudian la validez del sistema a través de pruebas unitarias y pruebas de interacción entre los diferentes dispositivos emisores y receptores.

7.1. Montaje del prototipo

En primer lugar es necesario realizar el montaje del hardware electrónico de las placas Arduino, esto es, conectar un módulo de recepción y emisión a cada una de las placas.

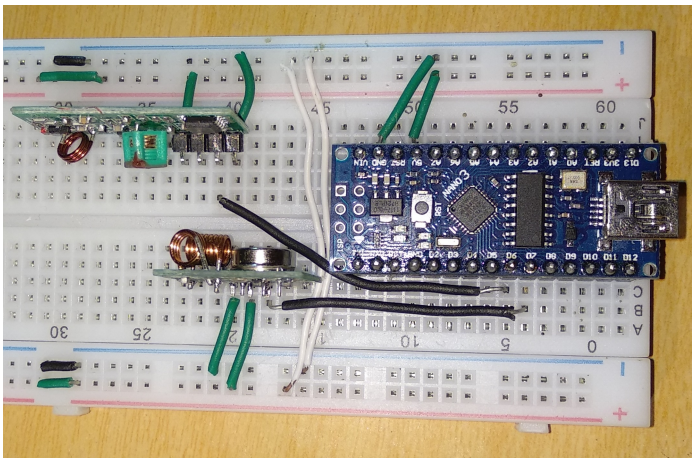

Figura 24: Vista en planta del montaje.

En la figura 24 se muestra el montaje de la electrónica en común de los tres tipos de nodo, es decir, simplemente la conexión de datos (cables negros) de los módulos RF. Así como su alimentación, que se realiza desde el Arduino (cables verdes).

A continuación se explican más detalladamente las diferentes pruebas a las que se ha sometido al prototipo.

7.2. Pruebas unitarias

El desarrollo del software del proyecto se ha realizado de una forma muy modular, como se describe en la sección 6.4. El software se basa en el uso constante de funciones muy simples pero que, en su conjunto, realizan las tareas. Las pruebas unitarias son los test aplicados sobre dichas funciones. A nivel individual se comprueba que cada una de ellas es capaz, en efecto, de completar su objetivo.

Al nivel de este proyecto, las pruebas unitarias se han ido realizando a lo largo del desarrollo del software puesto que el código se ha construido desde cero y siempre ha habido acceso al hardware de comunicación para probar dicho código en construcción.

El mayor impacto que han tenido las pruebas unitarias en este proyecto se ve reflejado en la gestión de las variables y, por tanto, la gestión del uso de la memoria. Las placas Arduino NANO cuentan sólo con 32 KB de memoria y es importante controlar como se esta aprovechando dicha memoria. Inicialmente, gran parte de las variables eran variables globales (práctica usual en los microprocesadores de Arduino para proyectos simples), sin embargo, no es óptimo para el uso de memoria y se acabó optando por utilizar las variables a nivel local dentro de las diferentes unidades de código.

7.3. Pruebas de características

Durante esta sección se pretende estudiar el comportamiento de los módulos de RF utilizados sometiéndolos a una serie de test simples como pueden ser distancia, velocidad de respuesta, o pérdida de paquetes.

7.3.1. Distancia y dirección de la comunicación

La distancia a la que los dispositivos son capaces de comunicarse es el parámetro más limitante para un sistema. Si el alcance de la comunicación no es el necesario será completamente inviable utilizar la mula de datos como solución al problema.

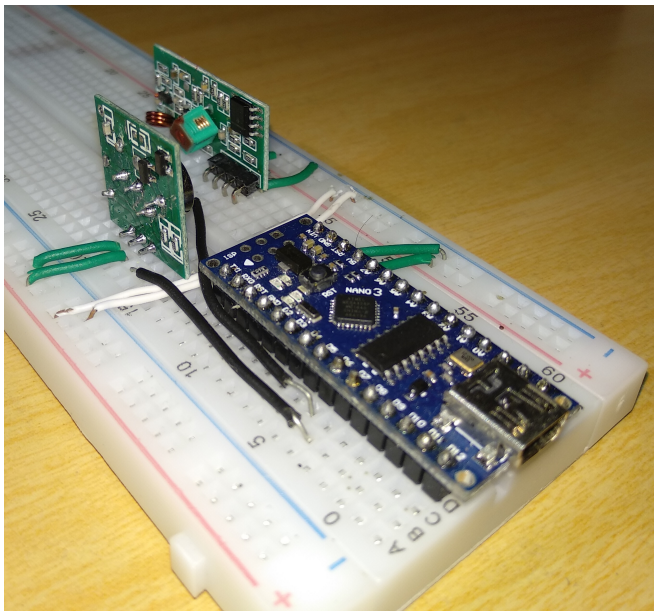

Figura 25: Vista lateral del montaje.

Durante las pruebas de distancia se ha podido apreciar que, como se indicaba en las características teóricas de los módulos la distancia puede alcanzar incluso $3,5$ metros.

Sin embargo, durante los test se ha observado que las distancias a las que los dispositivos se comunicaban variaba mucho en función de la geometría de los dispositivos. Esto indica que las antenas integradas en los módulos de comunicación son antenas direccionales. En la figura 25 se muestra la bobina que actúa como antena en el receptor.

En caso de integrar el dispositivo en un sistema real y diseñar un módulo que integre todos los componentes de este proyecto sería conveniente tener en mente este factor direccional de la onda electromagnética que se genera, con el fin de maximizar el alcance de los dispositivos.

7.3.2. Velocidad de respuesta y pérdida de paquetes

La velocidad a la que el dispositivo de recepción capta la información del emisor es otro factor importante a la hora de considerar el diseño. Se han realizado pruebas de velocidad y tiempo de respuesta.

Para la velocidad, se han enviado strings de diferentes tamaños desde un emisor hasta un receptor. Sin ningún tipo de delay (emisor) ni espera (receptor), el delay experimentado por el mensaje es de una media de 10 ms para un string de 5 caracteres y una media de 24 ms para 20 caracteres. Se puede afirmar que el tiempo de transmisión es rápido para string de tamaño pequeño.

Sin embargo, durante las pruebas de velocidad se observó que existía pérdida de paquetes, es decir, que los bloques de datos enviados no llegaban a su destino. Así que se optó por analizar el efecto de introducir un delay en el emisor de forma que el envío de datos fuera periódico, así como introducir un tiempo de espera de recepción de mensaje en el receptor. En la tabla de la figura 26 se muestran los porcentajes de paquetes recibidos. Cabe destacar que las pruebas se han realizado enviando mil paquetes.

Delay entre envio de mensajes	Tiempo de espera en el receptor					
	5 ms	10 ms	25 ms	50 ms	100 ms	200 ms
5 ms	100	100	100	100	99	100
10 ms	100	100	100	100	100	100
25 ms	98	99	100	100	100	100
50 ms	99	100	100	99	100	100
100 ms	100	100	97	100	96	100
200 ms	100	99	100	100	100	100

Figura 26: Tabla de los porcentajes de paquetes enviados con éxito.

En la tabla no se aprecia un patrón en los números, por lo que los tiempos de

espera y delays en el envío no son los causantes de la pérdida de paquetes. Si el tiempo de espera es mayor que el delay, el receptor simplemente espera el envío y no se pierde el paquete. Sin embargo, si el delay de envío es mayor que el tiempo de espera, es posible que el receptor realice escuchas de un emisor pero no encuentre ninguno y simplemente el paquete llegue en la siguiente escucha. Por estas razones, lo más probable es que las pérdidas de paquetes se deban a interferencias en la frecuencia utilizada y no a la configuración del hardware.

7.4. Pruebas del conjunto

A la hora de probar el código implementado en los tres tipos de nodos aparecieron muchos problemas. Principalmente causados por la sencillez de los módulos de radiofrecuencia. Como se explica en el apartado de hardware (sección 5.3.1) los módulos no cuentan con ningún protocolo integrado más que la capacidad de enviar y recibir bytes. Por ello, se implementaron diversas herramientas a través del software.

En primer lugar, se realizaron pruebas referentes a la comunicación uno a uno, para esquivar el modo broadcast predeterminado de los módulos. Enviando un array de datos simplificado (respecto a las capas de la sección 6) que contenía el identificador del destinatario y un mensaje, los dispositivos respondían de forma satisfactoria.

Adicionalmente, una de las pruebas consistía en tratar de comunicarse con un único receptor desde dos emisores simultáneamente. El comportamiento en este caso era errático y el receptor escuchaba los mensajes de los receptores de forma impredecible. Por ello, se debe evitar esta situación con dos emisores y un único receptor.

Después se pasó a implementar las funciones encargadas de enviar el ACK (*acknowledgement*). Tras recibir un bloque de información, el receptor debe enviar de vuelta un mensaje de tipo ACK y así informar al emisor que el mensaje se ha recibido. Y, a su vez, el emisor debe esperar la respuesta ACK antes de continuar con la ejecución del código.

Durante las pruebas se encontró que los nodos lidian mal con los cambios de papeles, es decir, si un nodo está actuando como receptor y debe pasar a ser un emisor momentáneamente (para enviar el mensaje ACK) se debe introducir una pequeña espera para que el sistema tenga tiempo de responder. Esto retrasa el tiempo de la comunicación ligeramente. Además del hecho de que el receptor, tras enviar el ACK no tiene forma de saber si ese ACK ha sido recibido por el emisor original, creando un punto débil en el proceso de comunicación.

Sin embargo, a pesar de estos obstáculos a la hora de implementar el diseño, el prototipo consigue funcionar en los tres niveles propuestos por Intel. El nodo sensor es capaz de enviar información a la mula de datos y ésta a su vez es capaz

de comunicarse con el servidor. No obstante, la mula cuenta con un input en el que el usuario debe indicar en que dirección (de servidor a sensor o viceversa) está tratando de comunicar los datos, pues incialmente, al acercar la mula a los otros nodos ambos trataban de actuar tanto de emisor como de receptor y el comportamiento no era el deseado.

8. Conclusiones y futuras líneas de trabajo

En este último capítulo se va a realizar un análisis de todo el trabajo realizado en este proyecto a través de las conclusiones alcanzadas. Además, se va a comentar posibles tareas de cara a una futura mejora o ampliación del trabajo realizado en este proyecto.

8.1. Conclusiones

Las conclusiones obtenidas durante el estudio y desarrollo de las mulas de datos de este trabajo son:

- Los objetivos propuestos se han completado con éxito. Se ha logrado establecer una comunicación bidireccional entre los nodos de tipo sensor y los servidores a través de una mula de datos. Además, se han realizado pruebas que determinan la validez del sistema en términos del hardware utilizado y el software desarrollado.

- Durante el estudio del estado del arte se ha comprobado que existen diferentes proyectos reales que emplean mulas de datos. Estos proyectos utilizan una estructura generalista que permite que la mula de datos funcione en distintos entornos de monitorización. Se ha visto entonces que hay disponibles diferentes alternativas para realizar un proyecto de estas características. No obstante, uno de los requisitos para el desarrollo de este proyecto era utilizar el hardware de Arduino y el IDE dedicado a dicho hardware, el cual ha funcionado satisfactoriamente.

- En este trabajo se han cumplido todos los objetivos establecidos, consiguiendo diseñar e implementar las funcionalidades de los tres niveles de las mulas de datos. Sin embargo, sería muy interesante desarrollar funcionalidades adicionales en otros trabajos de fin de grado como las propuestas en la sección de trabajos futuros (sección 8.2).

- La tecnología Arduino facilita en gran medida el desarrollo de aplicaciones, pues tanto las licencias de hardware como las de software son libres. Además, existe una gran comunidad de usuarios activa que puede ser de gran ayuda a la hora de desarrollar nuevas aplicaciones. La documentación es amplia y el acceso a ella es sencillo, principalmente por esta extensa base de usuarios. La tecnología Arduino ha resultado ser una opción adecuada, barata y versátil para el desarrollo de este trabajo.

- Durante el proyecto se han utilizado módulos RF conectados a las placas Arduino NANO. Sin embargo, estos módulos no son la opción más adecuada

para transmitir información de forma inalámbrica puesto que la sencillez de la librerías de comunicaciones complican demasiado el desarrollo código y no es conveniente a la hora de desarrollar sistemas de comunicaciones complejos.

- La plataforma de desarrollo ideal para este problema se basa en la utilización de ordenadores con el sistema operativo Mac OS. Las placas Arduino utilizadas son versiones baratas de las placas originales de Arduino y utilizan drivers propietarios. Durante el desarrollo de la aplicación se han detectado problemas de compatibilidad entre dichos drivers y las versiones IDE de Arduino cuando se desarrolla en ordenadores con el sistema operativo Windows 10. Estas incompatibilidades no se han presentado cuando las estaciones de desarrollo son máquinas dotadas del sistema operativo Mac OS. Debido a estas incompatibilidades el desarrollo de este proyecto se ha realizado utilizando un ordenador Mac Book Pro en el que el IDE de Arduino funciona sin problemas.

8.2. Trabajo futuro

Por último, las posibles líneas futuras de trabajo para este proyecto se presentan a continuación:

- Analizar posibles escenarios donde utilizar el proyecto y el comportamiento de la mula de datos en diferentes situaciones industriales. Incluso considerando la ampliación de la plataforma hacia el uso de varias mulas de datos simultáneamente en una red. Expandiendo la recolección de datos al conseguir que estas mulas de datos se comuniquen entre sí.

- Aplicar técnicas de ahorro energético para optimizar el tiempo de vida autónomo de los diferentes nodos.

- Implementar un módulo de memoria adicional (por ejemplo, un lector de memorias SD) para las placas Arduino para que éstas cuenten con mayor capacidad de almacenaje de datos. Es una alternativa utilizar placas de mayor tamaño (y por tanto mayor consumo) que además permitiría poder tener los datos en un medio portátil (tarjeta SD, USB, etc.).

- Integrar el uso de una base de datos MySQL para el nodo servidor. Junto a un gestor gráfico que permita al usuario realizar operaciones con toda la información almacenada en el Servidor.

- Añadir un nivel de seguridad adicional utilizando un sistema de cifrado para los mensajes enviados por los nodos, en caso de que la aplicación requiera este nivel de privacidad.

- Mediante el uso de técnicas de inteligencia artificial, aumentar la capacidad del sistema de actuar ante diversas situaciones, permitiendo a los nodos incluso alterar su propia configuración para optimizar la monitorización y la transmisión de datos.

- Desarrollo de servicios web que permitan acceder a los módulos desarrollados para enviar parámetros de configuraciones y comandos de operación.

9. Bibliografía

Referencias

[1] Virtualwire library.

[2] Lenguaje de programación de Arduino, estructura de un programa, Mar 2015.

[3] Programming arduino uno in pure c, Jun 2015.

[4] A. Chakrabarti, A. Sabharwal, and B. Aazhang. "using predictable observer mobility for power efficient design of sensor networks, 2003.

[5] A. S. Pentland, R. Fletcher, and A. Hasson. Daknet: Rethinking connectivity in developing nations, 2004.

[6] Arduino.cc. Arduino - arduinoisp. *Arduino Reference*, Jan 2018.

[7] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan, and S. Madden. "cartel: A distributed mobile sensor computing system, 2006.

[8] Miguel Baza Cuñado. Diseño e implementación de una plataforma de monitorización con mulas de datos para una red de motas dispersas sun spot. B.S. thesis, 2011.

[9] CircuitsToday. Arduino nano pinout and schematics - complete tutorial with pin description. *Electronic Circuits and Diagrams-Electronic Projects and Design. http://www.circuitstoday.com/arduino-nano-tutorial-pinout-schematics*, Aug 2018.

[10] D. Simon C. Cifuentes D. Cleal J. Daniels and D. White. "javatm on the bare metal of wireless sensor devices - the squawk java virtual machine. *http://labs.oracle.com/projects/squawk/docs/vee06-squawk.pdf*, Junio 2006.

[11] Designthemes. ¿cuánto consume arduino?

[12] Mario Di Francesco, Sajal K Das, and Giuseppe Anastasi. Data collection in wireless sensor networks with mobile elements: A survey. *ACM Transactions on Sensor Networks (TOSN)*, 8(1):7, 2011.

[13] T.M. Egyedi and D.C. Mehos. *Inverse Infrastructures: Disrupting Networks from Below*. Edward Elgar, 2012.

[14] G. Anastasi, M. Conti, and M. Di Francesco. Data collection in sensor networks with data mules: An integrated simulation analysis, 2008.

[15] Microchip Technology Inc. Atmega328p. *https://www.microchip.com/wwwproducts/en/ATmega328P*.

[16] J. Soares, M. Franceschinis, R. Rocha, W. Zhang, and M. Spirito. Opportunistic data collection in sparse wireless sensor networks, 2010.

[17] Philo Juang, Hidekazu Oki, Yong Wang, Margaret Martonosi, Li Shiuan Peh, and Daniel Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet. *SIGARCH Comput. Archit. News*, 30(5):96–107, October 2002.

[18] L. Levison and W. Thies and S. Amarasinghe. Providing web search capability for low-connectivity communities. In *IEEE 2002 International Symposium on Technology and Society (ISTAS'02). Social Implications of Information and Communication Technology. Proceedings (Cat. No.02CH37293)*, pages 87–91, June 2002.

[19] Luis Llamas. Comunicación inalámbrica en arduino con módulos rf 433mhz. *https://www.luisllamas.es/comunicacion-inalambrica-en-arduino-con-modulos-rf-433mhz/*, Oct 2017.

[20] Enrique Molinelli Fernández. Redes inalámbricas de sensores. nueva tecnología para aplicaciones navales., 05 2010.

[21] Museo Informática. Historia de las redes inalámbricas, 2010.

[22] Unkyu Park and John Heidemann. Data muling with mobile phones for sensornets. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, SenSys '11, pages 162–175, New York, NY, USA, 2011. ACM.

[23] Unkyu Park and John Heidemann. Data muling with mobile phones for sensornets. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, SenSys '11, pages 162–175, New York, NY, USA, 2011. ACM.

[24] A. Pentland, R. Fletcher, and A. Hasson. Daknet: rethinking connectivity in developing nations. *Computer*, 37(1):78–83, Jan 2004.

[25] Boselin Prabhu, Sophia Sudhir, P D Manivannan, S Nithya, and R Mahalakshmi. A research on decentralized clustering algorithms for dense wireless sensor networks. 57:975–8887, 11 2012.

[26] Arduino Uno Rev3. Arduino nano. *https://store.arduino.cc/usa/arduino-nano*.

[27] R. C. Shah, S. Roy, S. Jain, and W. Brunette. Data mules: modeling a three-tier architecture for sparse sensor networks. In *Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications, 2003.*, pages 30–41, May 2003.

[28] T. E. Starner. Wearable computing for the developing world. *IEEE Pervasive Computing*, 4:87–91, 07 2005.

[29] W. Brunette, D. Hoke, and J. Jenks. Proyect mule., 2002.

[30] C. Poellabauer W. Dargie. *Fundamentals of wireless sensor networks.* Wiley, 2010.

[31] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient mac protocol for wireless sensor networks, 2002.

[32] X. Shen, Z. Wang, and Y. Sun. Wireless sensor networks for industrial applications. 2004.

[33] Zahid, University of Southern California. Ad-hoc vs infrastructure, 2006.

[34] Pei Zhang, Christopher M. Sadler, Stephen A. Lyon, and Margaret Martonosi. Hardware design experiences in zebranet. In *Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems*, SenSys '04, pages 227–238, New York, NY, USA, 2004. ACM.

10. Anexos

A. Marco Regulador

Con el boom tecnológico de las telecomunicaciones en los últimos 50 años, se ha generado un gran mercado económico empresarial alrededor de éstas. Todos los países han establecido leyes que regulan el uso de todo tipo de comunicaciones inalámbricas. La situación financiera que afecta profundamente a todos los países desarrollados y la necesidad de fomentar la inversión y la competencia son factores a tener en cuenta en el marco regulador.

La legislación actual en España está recogida en la publicación del Boletín Oficial del Estado de referencia BOE-A-2014-4950[9] publicada originalmente el 10/05/2014 y actualizada por última vez el 04/07/2018. Por otra parte, el software y hardware referente a Arduino es de uso público por lo que existe libertad a la hora de usarlo.

En cualquier caso, este proyecto es meramente un proyecto de investigación referente a las mulas de datos. La única legislación que se podría aplicar es aquella referente a la propiedad intelectual. No obstante, toda la información utilizada de otras publicaciones ha sido correctamente referenciada y puede ser usada por cualquiera que adquiera los derechos de publicación.

[9]https://www.boe.es/buscar/act.php?id=BOE-A-2014-4950&p=20180704&tn=1

B. Entorno Socio-Económico

Este proyecto se basa en el análisis de la viabilidad de una mula de datos aplicada a un entorno industrial. Por ello, este trabajo en si mismo no tiene mayor impacto que la confirmación de que la mula de datos es un método viable, a pesar de que Arduino no sea la mejor opción para implementarla. Sin embargo, en caso de llevar a cabo un uso real del proyecto se vería reflejado en el coste de la implantación de una infraestructura de telecomunicaciones.

A corto plazo, la mula de datos supone un ahorro tremendo, sin embargo, la velocidad de transmisión de la información es mucho menor que la de una infraestructura tradicional. Además, en términos de largo plazo el mantenimiento de baterías en números nodos además de en las mulas de datos supondría un mayor coste que en la infraestructura clásica. Por ello, es posible que para empresas pequeñas o proyectos que no sean muy prolongados en el tiempo, el sistema de mula de datos ofrece una alternativa muy competente.

B.1. Presupuesto

En primer lugar hay que aclarar que los costes explicados en esta sección no son completamente imputables a este proyecto pues la mayoría del material sigue siendo utilizable tras el proceso de desarrollo. Se han realizado una serie de estimaciones en referencia al coste, temporal y material, de los elementos utilizados en este proyecto.

- Un ordenador portátil Mac Book Pro de 2014 con el sistema operativo **OS X** *El Capitan* que no necesita licencia una vez adquirido el portátil. Tiene un procesador Intel Core Duo de 2,53 GHz, un disco duro SSD de 256 GB y 4 GB de RAM DDR3. Este equipo ha sido la herramienta principal del ingeniero para realizar los trabajos de selección del hardware, compra, desarrollo, prueba y redacción del proyecto.

- Las licencias de todos los programas utilizados son gratuitas. LaTeX es de licencia abierta, al igual que el IDE de Arduino.

- El hardware utilizado si tiene un coste asociado. Ha sido necesario adquirir tres placas Arduino, tres conjuntos de emisor-receptor de módulos de radiofrecuencia, elementos de electrónica básica (LED's, Resistencias, etc.) y tres placas protoboard donde realizar el montaje del prototipo.

La siguiente table resume el coste económico que ha supuesto la adquisición de lo listado anteriormente.

	Precio Unitario $[\frac{e}{unidad}]$	Unidades	Precio total $[e]$
Mac Book Pro	1400	1	1400
Licencia Mac OS	Gratuita	1	0
Placa Arduino NANO	2.35	3	7.05
Módulo RF (pack receptor + emisor)	1.15	3	6.25
Pack de componentes electrónicos básicos	7.00	1	7.00
Pack 3 placas protoboard	9.99	1	11.99
Total			1432.29

Tabla 3: Presupuesto de los costes materiales.

A continuación se detallan las actividades que se han llevado a cabo durante el desarrollo de este proyecto:

- **Selección y adquisición del hardware.** Incluyendo el proceso de investigación las alternativas para realizar telecomunicaciones entre Arduinos.
- **Estudio y análisis del marco que rodea a las mulas de datos.** Tiempo dedicado a la investigación de la situación que rodea a la tecnología de telecomunicaciones y el estado del arte de las mulas de datos.
- **Aprendizaje de Arduino.** Incluye tanto el montaje del hardware electrónico como el aprendizaje del lenguaje del software.
- **Aprendizaje de Virtual Wire y Telecomunicaciones.** Engloba el tiempo dedicado al estudio de las prácticas más comunes para telecomunicaciones en Arduino.
- **Desarrollo del diseño del proyecto.** Tiempo dedicado al diseño e implementación de la plataforma, incluyendo los contratiempos encontrados y las pruebas realizadas.
- **Depuración y corrección.** Engloba el tiempo consumido por las tareas de mejora de los fallos encontrados en la implementación.
- **Redacción de la memoria.** Tiempo dedicado a redactar la memoria final presentada para el trabajo de fin de grado.

La siguiente tabla muestra un resumen de las tareas realizadas y las horas dedicadas:

Tarea	Horas Empleadas
Selección y adquisición del hardware	10
Estudio y análisis del marco que rodea a las mulas de datos	30
Aprendizaje de Arduino	35
Aprendizaje de Virtual Wire y Telecomunicaciones	18
Desarrollo del diseño del proyecto	125
Depuración y correción	60
Redacción de la memoria	30
Total	308

Tabla 4: Listado de tareas y horas empleadas.

También hay que tener en consideración el sueldo medio de un Ingeniero Industrial se estima en $20\frac{e}{hora}$ por lo que 308 horas de trabajo corresponderían a $6160e$.

Adicionalmente, existen una serie de costes varios como son los costes de envío del hardware o el consumo energético que se estiman alrededor de $10e$ y $50e$ respectivamente.

Por último se deben tener en cuenta los costes asociados a los impuestos, es decir, el impuesto del IVA del 21 %. Así, el presupuesto final del proyecto tiene el siguiente aspecto:

Descripción	Coste [e]
Coste material	1432.29
Mano de obra	6160
Coste de ejecución	7592.29
Costes varios	60
Presupuesto de ejecución	7652.29
Impuestos (IVA 21 %)	1606.98
Presupuesto final	9259.27

Tabla 5: Presupuesto económico final

Como se aprecia en la anterior tabla, el presupuesto del proyecto realizado asciende a NUEVE MIL DOSCIENTOS CINCUENTA Y NUEVE CON VEINTISIETE CENTIMOS.